Triplet Embedding Model: Explanation and Evaluation

Junhyeok Kim

Department of Statistics
Seoul National University
junhyuki@snu.ac.kr

Abstract

In NLP, a mapping from text data to Euclidean space is crucial in modeling process. We implement triplet embedding model by using *(subject, verb, object)* triplet data extracted from Reuters news headlines. We show clear syntactic relationships among various triplet vectors. Triplet vectors tend to gather around similar verbs and make different clusters according to similarities of *subjects* and *objects*. Interestingly, if we increase the size of the model, existing clusters are divided again to more various actors or actions. Major weakness is that triplet vectors are sensitive to order, shape and length of their *subject-verb-object* because of model architecture and the use of average vectors for each triplet component. Especially, using average vector produces good results only when length of *subject-verb-object* are short. However, when triplet vectors have many words, it produces arbitrary results. Hence, tackling the problem of average vector can be the key to generate more sophisticated performance.

1 Introduction

1.1 Motivation and the goal

In Natual Language Processing (NLP), learning representation of text data by vectors is a crucial starting point for modeling. Many attempts have been made for words (Mikolov et al., 2013 and Pennington et al., 2014), phrases and even documents (Mikolov, 2014 and Dai et al., 2015). Some of them showed amazing results like word vectors, but others left challenges. In this paper, we applied embedding method of *(subject, verb, object)*, SVO triplet data in short, a type of *multi-relational data*.

We followed Ding et al. (2015) for the baseline model. The authors implemented *event* (since the *event* is in triplet form, we will call it as *triplet* in this paper) embedding model to Reuters news data and tried to draw meaningful results from the distribution of news events in vector space. We thought this work is very interesting since finding out revelant relationships among events represented in vector space could be very useful in NLP and other areas. Unfortunately, authors missed out many details and did not clearly show the results. Moreover, they skipped the evaluation of triplet vectors in the first work, and ended up showing only a dozen of samples in the following work.

Hence, we decided to thoroughly investigate triplet embedding model[1]. We found that the model clearly reflects the syntactic relationships among various triplets. At first, they formed large clusters around shared *verbs*. And then, they made small clusters depending on their *subject* and *object*. Interestingly, if we increase the size of the model, triplet vectors were segmented into smaller clusters and represented different actions. However, one important drawback is that triplet embedding model uses average vectors for each *subject*, *verb* and *object*. Because of dilution effect, if their length

[1] All details & codes can be found in https://github.com/junhyeok-kim/TripletEmbeddingModel

is long, it is hard to explain differences of triplet vectors in each cluster. Hence, dealing with the problem of average is one of the most significant factors for performance improvements. We will dicuss details and some methods to tackle the problem.
Our main contributions are as follows. (i) Unlike other structure data like WordNet, FreeBase, and YAGO, we used real data that are so messy and noisy. Accordingly, compared to other studies, types of verb are diverse and the number of words in entities is relatively long from two to as many as thirty. Hence, embedding sentences and finding meaningful relationships can be a tough task, but still very interesting. (ii) We analyzed triplet vectors in a broad range and showed strong syntactic and some interesting semantic relationship among SVO triplets. (iii) We discussed some drawbacks of the model and suggested several methods and directions.

1.2 Related studies

Multi-relational data consists of two entities and one relationship. It is called *multi-relational* since one enitity can have multiple relationships with other entities. They are important for Knowledge Base (KB) and can be extended to many areas, such as analysis of social networks or recommender system where entities and relations play a key role (Bordes et al., 2013).
Usually, modeling *multi-relational data* mainly focuses on *link prediction* or *knowledge completion* which find hidden relations between two entities. Some meaningful attempts can be found in Chen et al. (2013) and Bordes et al. (2013). Also, several related studies were discussed in Socher et al. (2013): (i) Distance Model (Bordes et al., 2011) which applies linear transformation of two entities with L1-norm in the simplest form; (ii) Single Layer Model that uses scores by adding one more parameter vector; (iii) Hadamard Model (Bordes et al,. 2012) that changes the score function of two entities by inner products. Additionally, Socher et al. (2013) proposed Neural Tensor Network (NTN) model. It is more generalized version of previous model since it considers interaction effects between two entities.
However, embedding *multi-relational data* in Euclidean space is not widely adopted. Ding et al. (2015, 2016) proposed a triplet embedding model using NTN function and implemented end-to-end evaluation by using triplet vectors to predict stock prices. This model is basically a special case of Recursive Neural Tensor Network (RNTN) developed by Socher et al. (2013). One difference is that RNTN was done on word levels for sentiment analysis, instead of *subject-verb-object* level.
We think that triplet embedding model gave different point of view for embedding sentences or *multi-relational data*. Hence, in the paper, we aim to show all the details of triplet embedding process as much as possible. We describe the way of embedding SVO triplet data into numeric vectors and show some syntatic and semantic relations in a broad investigation.
The outline of the paper is as follows. In Section 2, we describe the triplet embedding model and the ways in which it can be developed. In Section 3, we explain how we preproceesed Reuters news datasets. In Section 4, we show experiments of several models and some interesting qualitative analysis using a nonlinear dimension reduction technique. Finally, in section 5, we discuss some important issues that we confronted in triplet embedding works.

2 Subject-Verb-Object Triplet Embedding Models

2.1 Baseline model

Triplet embedding model started with decomposing sentences into *subject*, *verb* and *object*. In terms of language structure, English sentence is in *subject-verb-object* order. So, the *verb* is at the center of sentence. For the baseline model, we referred Ding et al. (2015) where *event* embedding model is introduced. Here, *event* is same with *subject-verb-object* triplet. In Figure 1, the authors suggested the way of combining *subject-verb*, *verb-object*, and the results of the two, using bilinear form. It incorporates the information of sentence structure.
Basically, the base function of the model is called *Neural Tensor Network* (NTN) function (Socher et al., 2013). Unlike standard feedforward neural network, it considers the interaction effects between two entities e_1 and e_2:

$$g(e_1, e_2) = f\left(e_1^\top \boldsymbol{T}^{[1:k]} e_2 + \boldsymbol{W_1}^\top e_1 + \boldsymbol{W_2}^\top e_2 + \boldsymbol{b}\right) \tag{1}$$

where $\boldsymbol{T}$ is a tensor, $\boldsymbol{W_1}, \boldsymbol{W_2}$ is a weight matrix, $\boldsymbol{b}$ is a bias vector and f is an activation function.

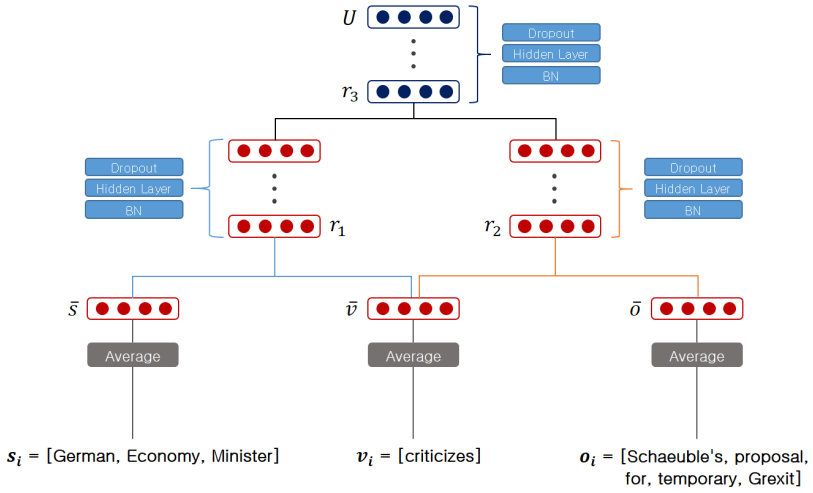


Figure 1: Triplet embedding model with some hidden layers

In mathematical formulation, there are *subject-verb-object* triplet data extracted from sentences.

$$(\boldsymbol{s}_i, \boldsymbol{v}_i, \boldsymbol{o}_i), \quad i = 1, \ldots, N \tag{2}$$

where $\boldsymbol{s}_i = (s_{1i}, s_{2i}, \cdots, s_{si})$, $\boldsymbol{v}_i = (v_{1i}, v_{2i}, \cdots, v_{vi})$, $\boldsymbol{o}_i = (o_{1i}, o_{2i}, \cdots, o_{oi}) \in \mathbb{R}^d$ and d is the dimension of word vectors. And we average *subject*, *verb*, *object*, respectively.

$$(\bar{s}_i = \sum_{j=1}^{S} s_{ji}/S, \;\; \bar{v}_i = \sum_{j=1}^{V} v_{ji}/V, \;\; \bar{o}_i = \sum_{j=1}^{O} o_{ji}/O) \tag{3}$$

In first, we combined $\bar{s}_i$ and $\bar{v}_i$ by using tensors and weight matrices to consider the interaction effects. Secondly, $\bar{v}_i$ and $\bar{o}_i$ are combined in the same way. And again, their output vector r_{1i} and r_{2i} are combined in the same way. For the sake of convenience, we dropped bias terms in equations.

$$r_{1i} = tanh(\bar{s}_i^\top \boldsymbol{T_1}^{[1:k]} \bar{v}_i + \boldsymbol{W_1}^\top [\bar{s}_i \oplus \bar{v}_i]) = g(\bar{s}_i, \bar{v}_i) \in \mathbb{R}^k \tag{4}$$

$$r_{2i} = tanh(\bar{v}_i^\top \boldsymbol{T_2}^{[1:k]} \bar{o}_i + \boldsymbol{W_2}^\top [\bar{v}_i \oplus \bar{o}_i]) = g(\bar{v}_i, \bar{o}_i) \in \mathbb{R}^k \tag{5}$$

$$u_i = tanh(r_{1i}^\top \boldsymbol{T_3}^{[1:k]} r_{2i} + \boldsymbol{W_3}^\top [r_{1i} \oplus r_{2i}]) = g(r_{1i}, r_{2i}) \in \mathbb{R}^l \tag{6}$$

where $\boldsymbol{T_1}^{[1:k]}, \boldsymbol{T_2}^{[1:k]} \in \mathbb{R}^{k \times d \times d}$, $\boldsymbol{W_1}, \boldsymbol{W_2} \in \mathbb{R}^{2d \times k}$, $\boldsymbol{T_3}^{[1:k]} \in \mathbb{R}^{l \times k \times k}$, $\boldsymbol{W_3} \in \mathbb{R}^{2k \times l}$ and $\oplus$ is a vector concatenation. And $u_i \in \mathbb{R}^l$ is the *triplet vector* where l is the dimension of it. Lastly, for the learning of triplet vectors, the score of a triplet is computed as follows:

$$score_i = f(\boldsymbol{\mu}^\top u_i) \in \mathbb{R} \quad \text{where} \;\; f = tanh \tag{7}$$

And we generate some corrupted triplets by randomly permutating the *subject* (simply changing the notation, $\bar{s}_i \rightarrow \bar{s}_i^c, \;\; c = 1, \cdots, C$) for each triplet and compute scores in the same process.

$$score_i^c = f(\boldsymbol{\mu}^\top u_i^c) \in \mathbb{R} \quad \text{where} \;\; f = tanh, \;\; c = 1, \cdots, C \tag{8}$$

We used *contrastive max margin loss* function to maximize the distance between real triplet's score and corrupted triplet's score.

$$L(\mathbf{\Phi}) = \sum_{i=1}^{N} \sum_{c=1}^{C} \max\{0, \alpha - score_i + score_i^c\} + \lambda \|\mathbf{\Phi}\|_2^2 \tag{9}$$

where $\mathbf{\Phi} = \{\boldsymbol{T_1}, \boldsymbol{T_2}, \boldsymbol{T_3}, \boldsymbol{W_1}, \boldsymbol{W_2}, \boldsymbol{W_3}, \boldsymbol{\mu}\}$ is the set of parameters in the model, N is a sample size, C is the number of corrupted samples, λ is the penalty parameter and α is a fixed margin. For training, we applied standard mini-batch gradient descent algorithm below.

Algorithm 1: Training algorithm of triplet embedding model

Data: triplet data: $\mathcal{T} = (\mathcal{T}_1, \cdots, \mathcal{T}_N)$ *where* $\mathcal{T}_i = (\bar{s}_i, \bar{v}_i, \bar{o}_i)$
corrupted triplet data: $\mathcal{T}^c = (\mathcal{T}_1^c, \cdots, \mathcal{T}_N^c)$ *where* $\mathcal{T}_i^c = (\bar{s}_i^c, \bar{v}_i, \bar{o}_i)$
Result: trained triplet embedding model

```
initialization;
while training loss ≤ stop criterion do
    Randomly split the training set into mini-batch size B;
    for b ← 1 to B do
        Φ_new = Φ_old − stepsize × Σ_j ∇L_j (Φ_old);
    end
end
```

2.2 Incorporating word vectors as learning parameters

For the baseline model, each word vector in *subject-verb-object* are used as input data. However, they can be learning parameters. suppose that $\boldsymbol{E}$ is a word matrix for *subject-verb-object* triplets. For example, 'German', 'government', 'bond', 'sale' are words in *subject*. Then, each word is encoded as a one-hot vector and look-up its word vector from word matrix $\boldsymbol{E}$ like:

$$(\boldsymbol{E}e_{1i}^s, \boldsymbol{E}e_{2i}^s, \cdots, \boldsymbol{E}e_{si}^s) = (s_{1i}, s_{2i}, \cdots, s_{si}) \text{ where } \boldsymbol{E} \in \mathbb{R}^{d \times N_w}, \quad e_{ji}^s \in \mathbb{R}^{N_w} \tag{10}$$

where e_{ji}^s is a one-hot vector for the *subject* and N_w is the number of unique words in SVO. In this way, each word in *verb* and *object* is also processed and remaining steps are the same for the baseline model. One difference is that word vectors can be changed in the direction of minimizing loss during the training process.

2.3 Some modified loss function

Actually, decision on which triplets need to be closed or not is the core. According to the form of loss function, the results can be quite different. Hence, we propose two loss functions and want to compare them with *constrastive max margin loss*.

2.3.1 Dynamic margin loss function

One drawback of the baseline model is that it uses fixed margin to make a distance between corrupted tirplet and real triplet. It means that corrupted but similar triplet would have a same distance with corrupted but dissimilar triplet. Hence, we thought that assigning different margins for each triplet might be effective. To incorporate this idea, we suggest *dynamic margin* that uses the similarities of word vectors. This approach has low cost to implement and also contains the information of corpus's context.
To do that, we calculate the cosine distance for each *subject*, *verb*, *object* between pairs of triplets like:

$$\boldsymbol{d}(\bar{s}_i, \bar{s}_j) = 1 - \frac{\langle \bar{s}_i, \bar{s}_j \rangle}{\|\bar{s}_i\|_2 \|\bar{s}_j\|_2} \tag{11}$$

To avoid dilution effect of average, we simply add them and apply exponential function to amplify the degree of the similarity.

$$\text{dm} = \exp(\boldsymbol{d}(\bar{s}_i, \bar{s}_j) + \boldsymbol{d}(\bar{v}_i, \bar{v}_j) + \boldsymbol{d}(\bar{o}_i, \bar{o}_j)) \tag{12}$$

Hence, final form is:

$$L(\mathbf{\Phi}) = \sum_{i=1}^{N}\sum_{c=1}^{C} \max\{0, \text{dm} - score_i + score_i^c\} + \lambda\|\mathbf{\Phi}\|_2^2 \tag{13}$$

where $\mathbf{\Phi} = \{\boldsymbol{T_1}, \boldsymbol{T_2}, \boldsymbol{T_3}, \boldsymbol{W_1}, \boldsymbol{W_2}, \boldsymbol{W_3}, \boldsymbol{\mu}\}$

2.3.2 Triplet loss function

Triplet loss is introduced by Schoroff et al.(2015) for face recognition. Suppose there is a certain person's image (anchor). Then it should be closer to same person's image (positive), but farther to other person's image (negative). Hence, in the learning process, authors used *triplet loss* to minimize the distance between an anchor and a positve, compared to between an anchor and a negative. For our triplets, we chose original SVO triplet as an anchor, *subject*-corrupted SVO as an positive, and *object*-corrupted SVO as a negative. This comes from the thought that different *subjects* with same *verbs* and *objects* might be similar and same *subjects* with same *verbs* but different *objects* have quite different meanings. Hence, the final form of *triplet loss function* is below:

$$L(\mathbf{\Phi}) = \sum_{i=1}^{N}\sum_{c=1}^{C} \max\left\{0, \alpha + \|score_i - score_i^{sc}\|_2^2 - \|score_i - score_i^{oc}\|_2^2\right\} + \lambda\|\mathbf{\Phi}\|_2^2 \tag{14}$$

where sc is *subject*-corrupted, oc is *object*-corrupted, and $\mathbf{\Phi} = \{\boldsymbol{T_1}, \boldsymbol{T_2}, \boldsymbol{T_3}, \boldsymbol{W_1}, \boldsymbol{W_2}, \boldsymbol{W_3}, \boldsymbol{\mu}\}$

3 Data

3.1 Gathering news data

In the real world, a lot of events occur and most of them are recorded in the news. Hence, news articles are relatively convenient sources to collect data. We searched several news websites and chose *Reuters* among them, since they provide *Archive*[2] which organizes massive articles in chronological order. We used `BeautifulSoup` library for web scraping, and collected 2,243,393 news articles from Jan 1, 2012 to July 1, 2018.

Year	2012	2013	2014	2015	2016	2017	2018	Total
# Articles	277,843	240,771	197,732	794,284	360,290	216,089	156,384	2,243,393

Table 1: Reuters news data

3.2 Data preprocessing

Raw news articles are somewhat *dirty* to be used in machine learning. Since each writer has different writing styles, it is difficult to have perfectly clean text data. Also, unlike plain texts, news contain a lot of puncutations (e.g. -&()={}:;[]), symbols, currencies (e.g. $, €), abbreviations (e.g. mln, bln, cbank), organizations (e.b. WB, IAEA), and people's names. What makes worse is, some of them are useless, but at the same time, others are useful depending on contexts. Hence, cleaning text data based on some rules inevitably contains errors because of contradictions. So, taking conservative point of view, we tried to minimize the errors as much as possible. To do so, we repeatedly selected random samples and carefully examined to find the types of errors. Also, by using *regular expressions*, we checked awkward word combinations.

Our text data preprocessing can be explained into two parts: headlines and bodies. For the headlines, as you can see in Figure 2, there are non-informative phrases (e.g. REFILE-, UPDATE-, :report) in the beginning or at the end of sentences. To find them among huge data, we came up with an idea that generallly covers many cases.

[2] `https://www.reuters.com/resources/archive/us/index.html`

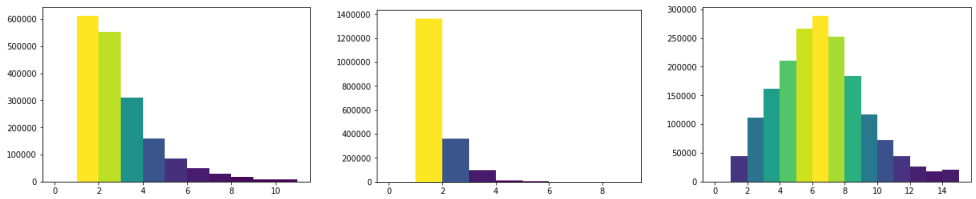
REFILE-UPDATE-U.S. seeks enhanced financial authority for Fed
Mariah Carey's "E=MC2" offers genre-crossing equation
Cell Therapeutics, Inc. Announces Filing of Form 10-K
Northwest gives Delta merger nod: source
Ex-NY Gov. Spitzer dodged DA probe by resigning
Metro-Goldwyn agrees entertainment projects in UAE
Nokia talks to major labels about music service: report
FACTBOX: Balkan candidates offer NATO leaner military muscle
REG-Invesco Eng.&Intnl;: Net Asset Value(s)
TEXT-Wesfarmers says chairman to stand down

Table 2: Some types of headlines

It is found that most of the meaningless parts of headlines were made of uppercase alphabets and could be distinguished by hyphens or colons. Therefore, we made it a rule that if the title contains one or more hyphens or colons, find shorter part divided by them and remove the part only when it is all in uppercases. For example, the first headline in Table 2 'REFILE-UPDATE-U.S. seeks enhanced financial authority for Fed' contains two hyphens. For the first hyphen, as the left part, 'REFILE', is shorter and all uppercased, it should be removed. Again, for the next hyphen, the left part 'UPDATE' is shorter and all uppercased, thus removed again. We obtain 'U.S. seeks enhanced financial authority for Fed' at last.
The bodies of news are also cleaned in similar way. As there are standard starting and ending phrases like 'SEOUL, Jan 4 (Reuters) $\cdots$ Reporting by $\cdots$', we deleted them. However, At the same time, some articles have no standard starting phrases. We assumed that they are less important because they are not in standard news format. As a result, the number of articles were reduced to 2,243,393.

3.3 Extract S-V-O triplet

We extracted SVO triplets from headlines since embedding all sentences could be tremendously noisy. Also, we can assume that headlines kept key information of articles. We first experiment *java*-based natural language software named *Stanford CoreNLP* and used annotators `tokenize`, `ssplit`, `pos`, `lemma`, `depparse`, `natlog`, `openie`. Unfortunately, this *openIE* (Open Information Extraction) extracted multiple SVOs and dropped some important words. So, it is difficult to distinguish which ones are appropriate data that bestly approximate the meaning of original sentences. Hence, we decided to use chunker trained on *CoNLL-2000* to find Verb Phrases, shortly VP, and simply extract *subjects* on the left side of VP and *objects* on the right side of VP.
And then, as shown in Figure 2, we plot histogram to find out how many words belong to *subject*, *verb* and *object*. We simply truncated the maximum length of them into 7, 3, 11 to eliminate rare triplets.



(a) Subject (b) Verb (c) Object

Figure 2: The length of *subject*, *verb* and *object*

Furthermore, triplets, which have wrong extracted and rarley appeared verbs, are eliminated to explain the model more clearly. Since our chunker is a kind of *supervised learning*, its performance depends on the datasets. *CoNLL-2000* data consists of Noun Phrase (NP, 51%), Verb Phrase (VP, 20%) and its sentences are relatively simple compared to our dataset. Indeed, in real news data, there are many abbreviations, uppercase of text for emphasizing it. They made the trained chunker to find wrong verb phrases. Nevertheless, one positive thing is that most parts of wrong extracted SVO turned out

meaningless headlines such as ‘ db x-trackers MSCI WORLD FINANCIAL TRN INDEX ETF Net Asset Value $\cdots$ ’. Hence, we tried to eliminate them as much as possible.

3.4 Word embedding

For estimation, text data should be converted some numeric data before modeling. Thus, choice of representation in SVO triplets affects the goal. We converted *subject-verb-object* into dense word vectors by using *word2vec* model. Actually, word vectors are already extensively used in natural language processing and other areas. Even many well-trained word vectors like *GoogleNews Vectors* or *GloVe Vectors* are publicly available. However, we trained our own word vectors on the headlines and bodies of articles, since the performance of word vectors depends on the corpus size, corpus type, and time periods. We expected that using our own trained word vectors can adaptively reflect the characteristics of SVO triplets. We tested *skipgram* and *CBOW* model and chose the former one for the reason of better semantic accuracy (Mikolov et al., 2013).

4 Experiments and Evaluations

4.1 Data descriptions

As shown in Table 3, training data is 806,055 headlines of news articles after preprocessing. Also, most common words are shown in Table 4 and the maximum length of *subject-verb-object* is 7, 3, 11, respectively. Our goal is to distribute triplet vectors containing the information of each headline in Euclidean space.

# Verbs	# Subjects	# Objects	# Training
2,745	305,480	731,446	806,055

Table 3: Data statistics

Subject (7)	[(’China’, 5394), (’U.S.’, 5206), (’Wall St’, 4983), (’Russia’, 3196), (’Fitch’, 2987), (’Obama’, 2660), (’European shares’, 2462), (’TSX’, 2286), (’EU’, 2112), (’Euro’, 1893)]
Verb (3)	[(’says’, 85250), (’sees’, 18023), (’reports’, 11283), (’announces’, 11234), (’Announces’, 11017), (’posts’, 9265), (’to buy’, 7681), (’raises’, 7386), (’rises’, 6928), (’falls’, 6893)]
Object (11)	[(’down’, 375), (’Quarterly Dividend’, 367), (’trading halt’, 331), (’estimates’, 322), (’change of accounting policy’, 300), (’2017 dividend payment’, 283), (’regulatory approval for private placement’, 278), (’Second Quarter 2015 Results’, 172), (’no dividend payment for 2017’, 167), (’up’, 164)]

Table 4: 10 most common words in *subject-verb-object*

4.2 Triplet embedding models

In experiments, we want to find some meaningful syntactic and semantic relationships among SVO triplets. We applied 4 models. (i) baseline model with some modifications; (ii) baseline model with incorporating word matrix $\boldsymbol{E}$ as an additional parameter; (iii) baseline model with dynamic margin loss function; (iv) baseline model with triplet loss function. However, for our experiment, appending word vectors into the learning process broke syntactic relationships what we expected. Hence, we will not cover this method in this paper.

For training, we used L1-penalty to all parameters except biases for regularization. Especially, we often experienced that value of each coordinate in triplet vectors are easily diverged to -1 or 1 through *tanh* function in the process of training. Hence, in final experiments, we applied *batch normalization* (Ioffe et al., 2015) for the inputs of *tanh* to reduce the training costs and also get some regularization effects for overfitting problem. Surprisingly, batch normalization not only stabilize the training

Models
Baseline (Ding et al. 2015)
Baseline + word matrix parameters
Baseline + dynamic margin loss
Baseline + triplet loss

Table 5: Triplet embedding models

process but also dramatically reduce the training time and improve the results. Moreover, we changed learning rate cyclically from minimum bound to maximum bound by following *Cyclical Learning Rate* (Smith, 2017) to avoid saddle points or bad local minimums.
Specifically, baseline model setting is (i) minimun learning rate $= 0.0000006$, maximum learning rate $= 0.000001$, (ii) dropout ratio $= 0.3$, (ii) batch normalization with learning parameters $\beta = 0$, $\gamma = 1$, (iii) word vector parameters initialized by trained skipgram model (iv) Other parameters initialized by uniform distribution $[-0.0025, 0.0025]$ (v) optimizer $=$ mini-batch gradient descent, (vi) $\lambda = 0.01$, (vii) margin $(\alpha) = 10$.
Also, since triplet embedding model is one of the *unsupervised learning*, we adopt stop criterion on the basis of the degree of training. In other words, We stopped training if the model's average epoch loss is less than 0.1 which means that about 90% of training triplets are learned compared to their corrupted triplets. Lastly, we expermented k from 60 to 200, l from 60 to 200, C from 3 to 7, and the number of hidden layers from 3 to 10 with different learning rates and initializations.

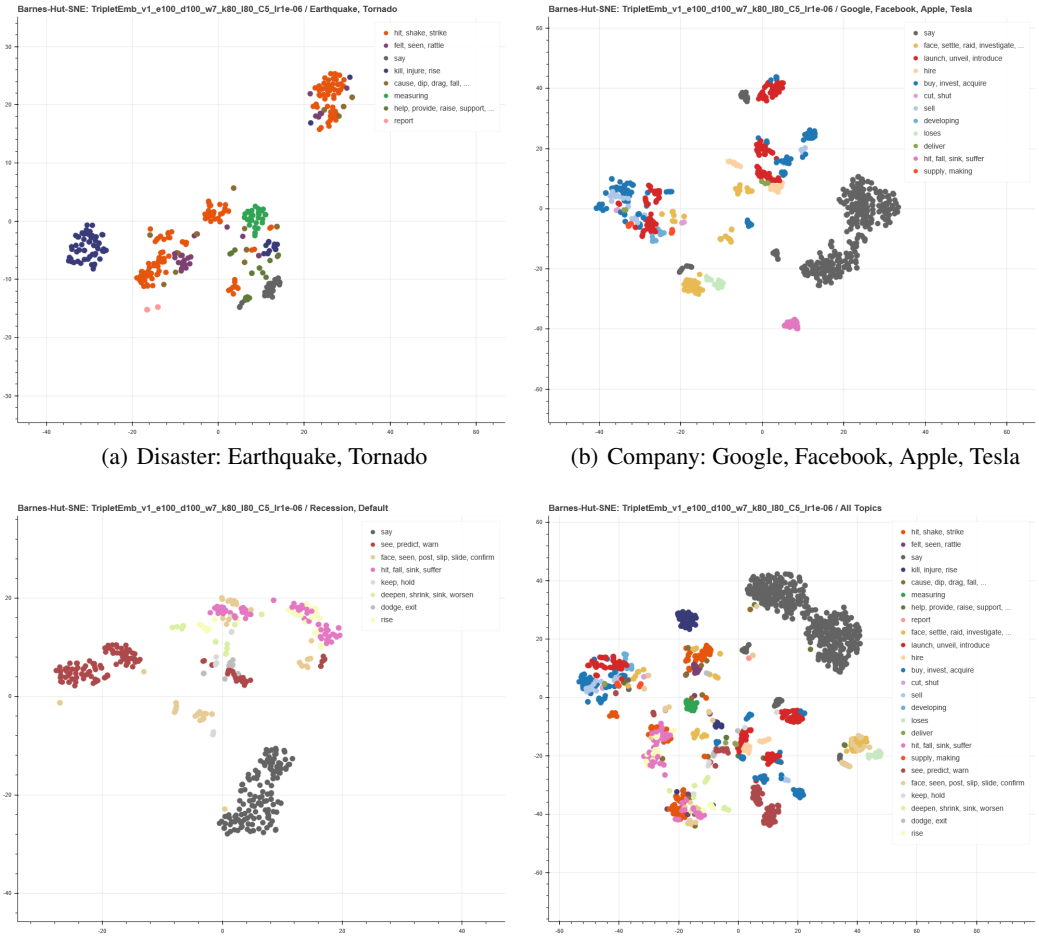
4.3 Qualitative Analysis

In qualitative analysis, we investigated the quality of triplet vectors in various aspects. We used `Multicore-TSNE`[3] for 2-D embedding of high dimensional vectors and `bokeh`[4] for interactive plots.

4.3.1 Selected topics: baseline model

Since the model iteratively combines *subject-verb-object*, we expect various events to be clustered around their verbs. Also, due to simlilarities of word vectors, triplets with different entities might be gathered differently in spite of same verbs. To confirm this conjecture, we made a testset that consists of several different topics in Appendix 1.
There are 8 topics with various events. First 2 topics are categorized into *Natural disaster*, next 4 topics are *Major companies* and the others are *Economy*. Within categories, we broke down several different events such as *acquisition*, *development*, *side-effect of recession* and *damage by earthquake*. We assigned different colors according to the types of verbs to find out whether similar events are gathered in same clusters or not. We implemented Barnes-Hut-SNE and visualized them in 2-D embedding plots.
We observed that triplet vectors have a clear syntactic relationships. As shown in Figure 3, SVO triplets sharing same verbs tend to be gathered in same cluster. For example, 'Nikkei edges down on stronger yen', 'Euro edges up from 16-mth low', and 'Wall Streets edges up as consumer sentiment jumps' have short distance. Also, we assigned the same color for verbs having similar meaning. For example, in Figure 3(a), *hit*, *shake* and *strike* are assigned to orange color. Even though they all mean the occurrence of natural disaster, they made different clusters due to the effect of word vectors.
However, triplet vectors sometimes could not show clear distinction among totally different entities such as *Google* and *Earthquake* so that distance between them is not that far. In other words, if they share a same verb, there is a possibility to be in the same cluster in spite of huge difference between their entities. And we think that this is one of the weaknesses of the model.
In addition, we found that the form of sentences can affect the distribution of triplet vectors. For example, in Figure 4(a), we selected some samples in active and passive forms; 'Tornadoes kill four people, $\cdots$' and 'At least two people killed in tornadoes $\cdots$'. The only change is the order of *subject* and *object*, but they made two different clusters because of model architecture. Lastly, one interesting point is that if we add more hidden layers, events tend to be divided into smaller clusters according to their entities. As we can see in Figure 4(b), in spite of the same verb *says*, several actors 'Google',

[3] `https://github.com/DmitryUlyanov/Multicore-TSNE`
[4] `https://bokeh.pydata.org/en/latest/`



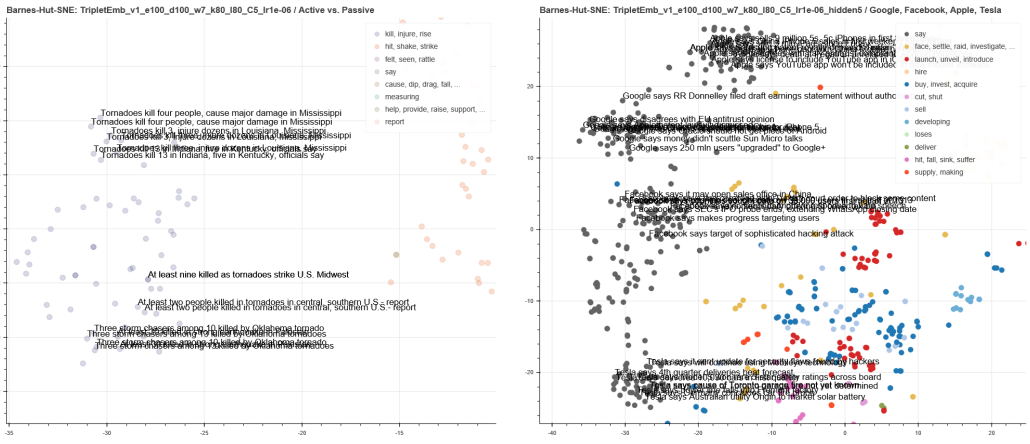
(a) Disaster: Earthquake, Tornado

(b) Company: Google, Facebook, Apple, Tesla

(c) Economy: Recession, Default

(d) All topics

Figure 3: Results of triplet vectors using *contrastive max margin loss function*. We trained baseline model with batch normalization and setting $d = 100$, $k = 80$, $l = 80$, $C = 5$, $\lambda = 0.01$, initial width$= 0.0025$, maximum learning rate $= 0.000001$



(a) Active form vs. passive form

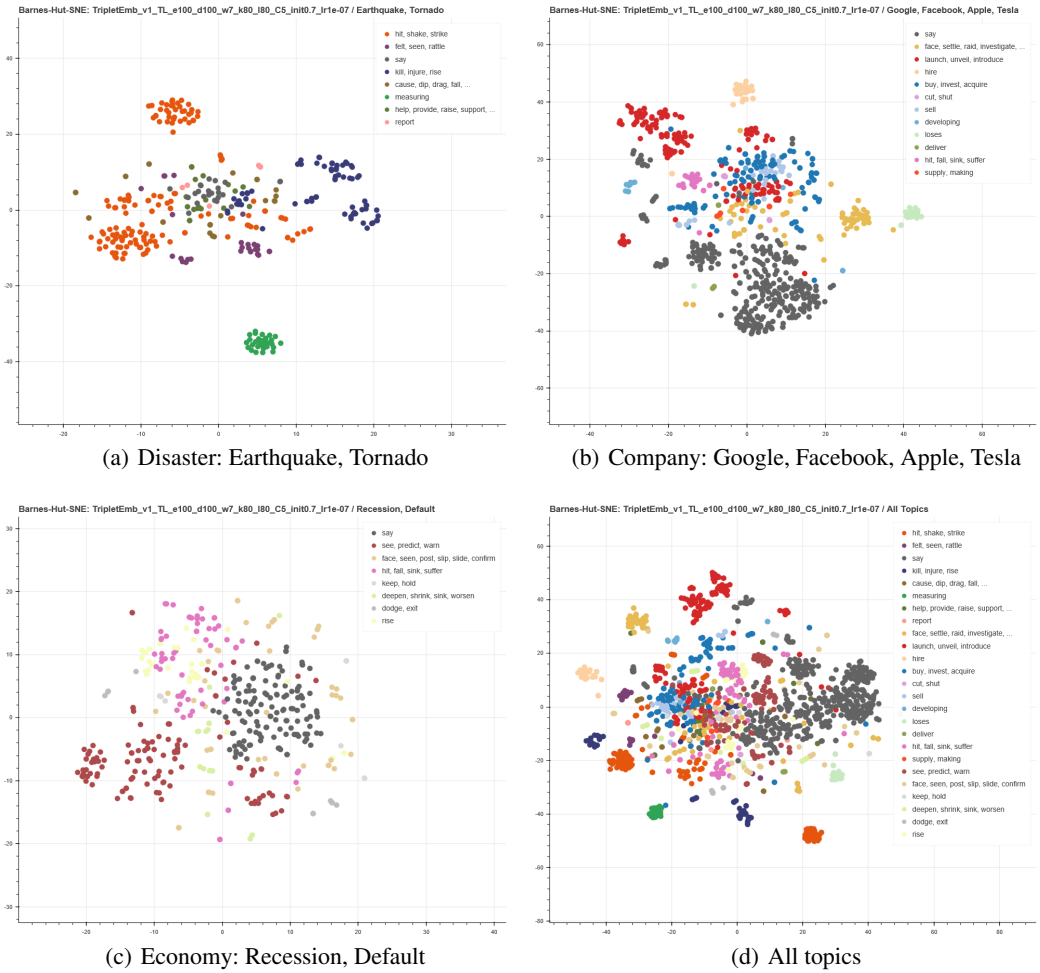
(b) Add 5 hidden layers

Figure 4: Some interesting cases

'Facebook', 'Apple' and 'Tesla' are recognized as different actions.

4.3.2 Selected topics: triplet loss

Using *triplet loss* produced different results compared to *contrastive max margin*. In Figure 5, it made more diverse and smaller clusters but at the same time, made other triplets dispersed. At the first appearance, its performance looks worse than the baseline model. However, if we look more closely, it showed very interesting results. In fact, *Triplet loss* makes SVO triplet and S^cVO triplet (corrupted *subject*) closer but SVO triplet and SVO^c triplet (corrupted *object*) farther. Natually, triplet vectors are divided into smaller clusters compared to the baseline model since triplets with different *object* will be additionally seperated. Unfortunately, our data have long *objects*. Since we average each word vector to ensure fixed dimension of triplet's components, *object* part are not guaranteed to maintain the similarities of two very different *objects*. In results, lots of triplets are dispersed in the center of plots.



(a) Disaster: Earthquake, Tornado

(b) Company: Google, Facebook, Apple, Tesla

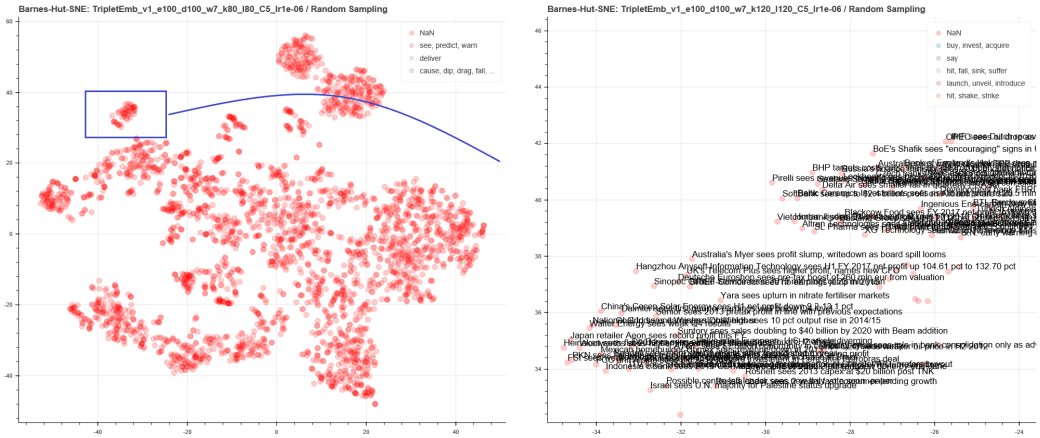
(c) Economy: Recession, Default

(d) All topics

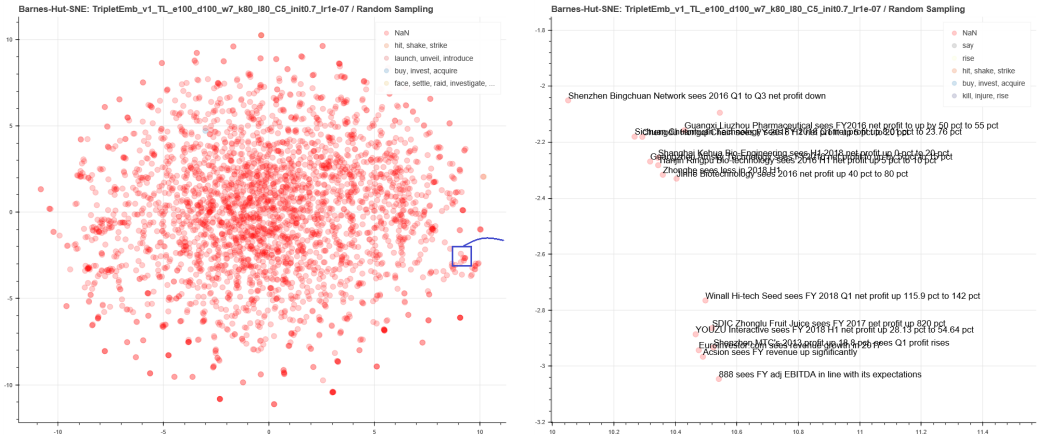
Figure 5: Results of triplet vectors using *triplet loss function*. We trained baseline model with batch normalization and setting $d = 100$, $k = 80$, $l = 80$, $C = 5$, $\lambda = 0.01$, initial width$= 0.7$, maximum learning rate $= 0.0000001$

4.3.3 Random sampling

For the concreteness, we selected 3,000 random samples and visualized in 2-D embedding plots. Not only for testset, but also for random samples, it showed clear syntactic relationships in Figure 6. There are bunch of clusters and we can think that they contained the information of similarity



(a) Random Sampling 3,000 (b) Entities sharing the verb 'sees'



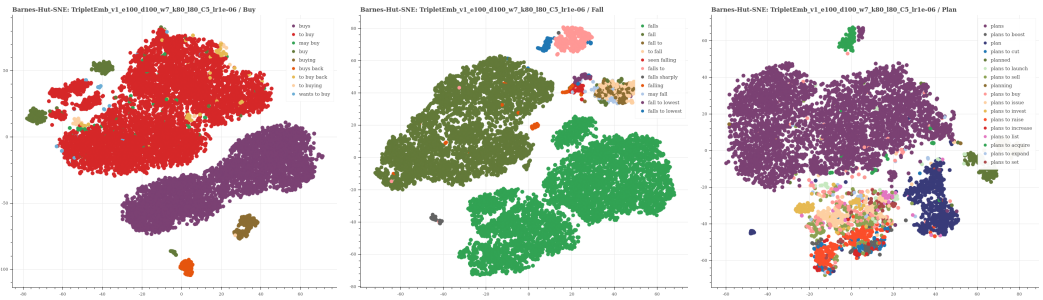
(c) Random Sampling 3,000 (d) Entities sharing the verb 'sees'

Figure 6: 2-D embedding of 3,000 random samples. Above two figures are from the baseline model and belows are using *triplet loss*

among various triplets. If we see them more closely in interactive plots, we can find that triplets gathered around their *verbs* and dispersed according to their *subjects* and *objects*. And for *triplet loss*, compared to the baseline model, it produced more sophisticated clusters in terms of *objects*.

4.3.4 Main verb and its verbs phrases

We found that the model is somewhat sensitive to the shape of verbs. For example, same entities with slightly different verbs like 'buy' and 'to buy' are not in the same cluster. To investigate the effect of the shape of verbs, we made a testset in Appneidx 2. There are three verbs, 'buy', 'fall' and 'plan' and phrases made with them. As shown in Figure 7, clusters are created differently according to usage of verbs in each cluster. We thought that the results come from average vector of the verb phrases. Hence, ('to buy', 'buys') or ('falls', 'fall to') can be different in terms of average. Obviously, using average word vectors ignore the order of words like ('fall to', 'to fall'). In addition, since we use word vectors, tenses, singular/plural and other grammatical differences affect the number of clusters like ('plan', 'plans', 'planned', 'planning'). Hence, if someone wants to ignore the effect of differences in verb phrases, we recommend to eliminate unnecessary preposition like 'to', 'of', 'by' or to lemmatize verbs, unless existence of preposition or other words have important information. It can be done in the preprocessing step of generating word vectors.



(a) ‘buy’ with similar verbs (b) ‘fall’ with similar verbs (c) ‘plan’ with similar verbs

Figure 7: 2-D embedding of some verbs with similar ones

5 Discussion

We saw many details of triplet vectors in various angles. In discussion, we shared the weakness of the model and compared loss functions. Also, we suggested the appropriate way of using triplet embedding model.

5.1 Weakness: averaging word vectors

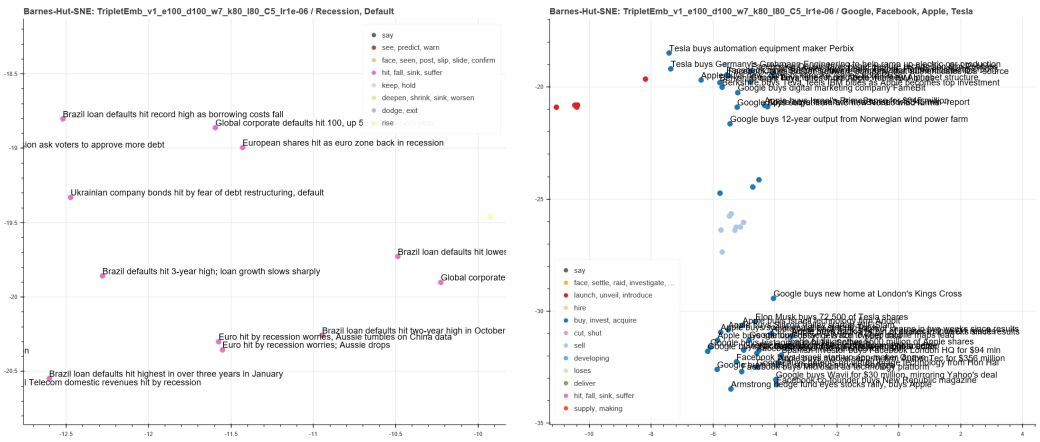
Since a lot of events occur everyday, it is not easy to distinguish them in vector space with reasonable distances among clusters. Also, triplet embedding model does not have a function to make a distance between clusters which are thought to be different. The distance depends only on the average similarity of word vectors. Using average vectors can be a good idea only when the length of *subject-verb-object* is short ($\leq$ 2 to 3 words). For example, as shown in Figure 8(a), most of *subjects* have ‘Brazil’, ‘loan’, ‘defaults’. Hence, triplets with the *subject* like ‘Brazil loan defaults’, ‘Brazil defaults’, and ‘Global corporate defaults’ are similar in terms of average vectors. In another example, ‘Korean court’, ‘Arizona court’, ‘Mississippi court’ are slightly different but they are all ‘court’. Average vector can reflect their similarity by sharing word ‘court’. In contrast, if length is long, averaging can produce arbitrary results. In Figure 8(b), there are bunch of triplets with long *object* part. They made two clusters with the form of sentences *company buys something*. However, it is very difficult to find proper explanation for it forming different clusters. This is the typical problem of average. Hence, for more reasonable performance, we should tackle this problem and find more robust way of recognizing entites that are truly similar. We think that one can consider LSTM or GRU cell’s last hidden states instead of average vectors to capture current and all previous information as much as possible.

5.2 Loss functions

After lots of experiments, we concluded that *dynamic margin loss* is not effective since it collapsed the existing syntactic relationship genereated by baseline model. However, *triplet loss* generated interesting results. Although it does not show beautiful local structures of triplet vectors compared to baseline model, we can find some more sophisticated segments. Unfortunately, due to the problem of average we discussed, we sometimes cannot see clear distinction among clusters since *object* part is too long.
We think that selecting loss functions is a matter of choice. According to the goal of project, they will show different results from intrinsic evaluation to end-to-end evaluation. For example, if we use triplet vectors which are trained on some text data to classify a lot of documents into very specific categories, then *triplet loss* can be the choice. If not, baseline model can be the choice for classifying into broader categories.

5.3 The direction of data preprocessing

As we saw in qualitative analysis, triplet embedding model is sensitive to order and shape of *subject-verb-object*. Hence, data analyst should decide the direction of preprocessing. For example, if little



(a) Short length of *subject* (b) Long length of *object*

Figure 8: Pros and cons of using average word vectors

differences of verb phrases can be ignored for the specific model, using stemmer, lemmatizer, or some user-defined functions would be a good solution. However, if some words such as 'may' or 'can' are important to discriminate the meaning of sentences, adding them is more appropriate action. Additionally, for the quality of triplet data, *Named Entity Recognition* (NER) can improve the accuracy of extracting *verbs*. Since there are a lot of organizations, person's names, and abbreviations, different representations of the same entity are not regarded as same one. For example, 'The Wall Street Journal' and a set of words 'The', 'Wall', 'Street', 'Journal' are quite different in terms of word vectors. Furthermore, 'WSJ' is also quite different from the average vector of the 4 different words. But they are all same entity indicating the name of company. Consequently, if we do not apply NER, triplets with various representation of *entities* will hide the true distribution of them in vector space. Hence, if we recognize an important entities by NER and converting them into unified words, they will have a similar distribution in vector space. And this can be one of the methods for minimizing the problem of average.

6 Conclusion

We showed very interesting results by investigating triplet embedding models in various aspects. Reasonably, model produced good syntactic regularity in terms of the combination of *subject*, *verb* and *object*. Moreover, we can find some meaningful relationships among triplets. However, there are many problems such as using average vector. For the long phrases, we need to find the way of capturing its tru meaning. We think that overcoming the method of average would produce very impressive results. We hope that following works tackle this weakness of triplet embedding model.

References

[1] Ding, X., Zhang, Y., Liu, T., & Duan, J. (2015). Deep learning for event-driven stock prediction. *Ijcai (pp. 2327-2333)*

[2] Ding, X., Zhang, Y., Liu, T., & Duan, J. (2016). Knowledge-driven event embedding for stock prediction. *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers (pp. 2133-2142)*

[3] Socher, R., Chen, D., Manning, C. D., & Ng, A. (2013). Reasoning with neural tensor networks for knowledge base completion. *Advances in neural information processing systems (pp. 926-934)*

[4] Bordes, A., Weston, J., Collobert, R., & Bengio, Y. (2011, August). Learning Structured Embeddings of Knowledge Bases. *AAAI (Vol. 6, No. 1, p. 6)*

[5] Bordes, A., Glorot, X., Weston, J., & Bengio, Y. (2014). A semantic matching energy function for learning with multi-relational data. *Machine Learning, 94(2), 233-259*

[6] Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., & Yakhnenko, O. (2013). Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems (pp. 2787-2795)*

[7] Schroff, F., Kalenichenko, D., & Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. *Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 815-823)*

[8] Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*

[9] Smith, L. N. (2017, March). Cyclical learning rates for training neural networks. *Applications of Computer Vision (WACV), 2017 IEEE Winter Conference on (pp. 464-472). IEEE*

[10] Tjong Kim Sang, E. F., & Buchholz, S. (2000, September). Introduction to the CoNLL-2000 shared task: Chunking. *Proceedings of the 2nd workshop on Learning language in logic and the 4th conference on Computational natural language learning-Volume 7 (pp. 127-132). Association for Computational Linguistics*

[11] Yang, B., Yih, W. T., He, X., Gao, J., & Deng, L. (2014). Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*

[12] Manning, C., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S., & McClosky, D. (2014). The Stanford CoreNLP natural language processing toolkit. *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations (pp. 55-60)*

[13] Hill, F., Cho, K., & Korhonen, A. (2016). Learning distributed representations of sentences from unlabelled data. *arXiv preprint arXiv:1602.03483*

Appendix

Appendix 1: Testset consists of 7 different topics

Category1	Category2	Category3	Verb Lists	# samples	Cluster Number	Palettes
Disaster	Earthquake	damage by earthquake	['killed', 'killing', 'kills' 'bears', 'injures', 'rises', 'rises to']	16	0	#393b79
		help for damage in earthquake	['Helps', 'Provides', 'Raises', 'to Support', 'Receives', 'approves', 'Responds', 'to help', 'to provide', 'Supports', 'donates', 'launches']	18	0	#637939
		side-effect of earthquake	['cause', 'causes', 'knocks', 'dips', 'drag', 'falls', 'tumbles', 'triggers']	17	0	#8c6d31
		earthquake felt	['felt', 'rattled', 'rattles', 'seen']	27	0	#7b4173
		earthquake hit	['hit', 'hits', 'shaken', 'shakes', 'strikes']	109	0	#e6550d
		earthquake measured	['measuring']	29	0	#31a354
		say	['say', 'says']	19	0	#636363
	Tornado	damage by tornado	['kill', 'killed', 'kills', 'injures', 'injured', 'rips', 'rises to']	54	1	#393b79
		side-effect of tornado	['cause', 'caused', 'causes', 'hail']	7	1	#8c6d31
		tornado hit	['hit', 'hits', 'slam', 'slams', 'strike', 'tears', 'sweep', 'touches']	39	1	#e6550d
		report	['reported', 'reports']	6	1	#ff9896
		say	['say', 'says']	5	1	#636363
Company	Google	say	['says', 'Says', 'announces']	159	2	#636363
		buy	['to buy', 'buys', 'to buying', 'to invest', 'invests', 'to acquire', 'acquires']	61	2	#1f77b4
		sell	['to sell']	7	2	#aec7e8
		launch	['launches', 'to launch', 'unveils', 'to unveil' 'aims', 'introduces']	50	2	#d62728
		face (e.g. lawsuit)	['faces', 'to face', 'settle', 'raid', 'investigating', 'to investigate', 'sues', 'fined']	38	2	#e7ba52
		hire	['hires']	16	2	#fdd0a2
		development	['developing']	7	2	#6baed6
		cut	['to cut', 'to shut']	7	2	#de9ed6
	Facebook	say	['says', 'Says', 'announces', 'reports']	62	3	#636363
		buy	['to buy', 'buys', 'to acquire', 'acquires', 'to build']	41	3	#1f77b4
		sell	['to sell', 'sells']	15	3	#aec7e8
		launch	['launches', 'to launch', 'unveils', 'expands', 'to expand', 'plans']	57	3	#d62728
		face (lawsuit, etc)	['faces', 'to investigate', 'sues', 'fined']	17	3	#e7ba52
		hire	['hires']	7	3	#fdd0a2
		development	['developing']	4	3	#6baed6
	apple	say	['says', 'announces']	65	4	#636363
		buy	['buys', 'to buying', 'to invest', 'invests', 'to acquire', 'acquires']	32	4	#1f77b4
		sell	['to sell']	9	4	#aec7e8
		launch	['launches', 'to launch', 'unveils', 'to unveil', 'introduces', 'plans']	43	4	#d62728
		face (lawsuit, etc)	['faces', 'settle', 'sues', 'fined']	36	4	#e7ba52
		hire	['hires']	8	4	#fdd0a2
		loses	['loses']	24	4	#c7e9c0
		development	['developing']	2	4	#6baed6
	Tesla	say	['says', 'Says', 'announces', 'posts', 'reports']	67	5	#636363
		shares movement	['fall', 'dive', 'drop', 'rise', 'hit', 'seen rising']	26	5	#e377c2
		buy	['to buy', 'buys', 'to invest']	14	5	#1f77b4
		unveil	['unveils', 'to unveil', 'introduces']	17	5	#d62728
		partner company	['to supply', 'making']	12	5	#fc4e2a
		deliver cars	['delivers', 'to deliver']	5	5	#8ca252
economy	recession	recession prospective	['sees', 'see', 'predicts', 'warns']	57	6	#ad494a
		say	['says']	42	6	#636363
		side effect of recession (e.g. profit, share price down)	['hit', 'hits', 'falls', 'fall', 'sink', 'suffer']	33	6	#e377c2
		current (negative or positive)	['faces', 'seen', 'posts', 'slips', 'slides', 'confirms', 'confirmed']	39	6	#e7cb94
		more worse	['deepens', 'shrinks', 'sinks', 'worsens']	18	6	#d9f0a3
		temporary avoid recession	['dodges', 'exits']	10	6	#bdbdbd
		meditation	['keeps', 'holds']	8	6	#d9d9d9
	default	recession prospective	['sees', 'see', 'warns']	60	7	#ad494a
		say	['says']	94	7	#636363
		rise	['rise', 'rises', 'rise to', 'seen rising']	32	7	#f7fcb9
		side effect of default (e.g. profit, share price down)	['hit', 'hits', 'falls', 'fall']	31	7	#e377c2
		current (negative or positive)	['faces', 'seen', 'slips', 'slides']	10	7	#e7cb94
Total	-	-	-	1,688	-	26 colors

Appendix 2: Testset consists of verbs and their similar verbs

Verb	Derivative Verbs	# samples	Cluster Number	Palettes
Buy	agrees to buy	324	0	#393b79
	buy	15,074	0	#637939
	buying	361	0	#8c6d31
	buys	5,388	0	#7b4173
	buys back	170	0	#e6550d
	may buy	104	0	#31a354
	offers to buy	240	0	#636363
	plans to buy	184	0	#ff9896
	says to buy	205	0	#1f77b4
	seeks to buy	41	0	#aec7e8
	to buy	8,902	0	#d62728
	to buy back	120	0	#e7ba52
	to buying	74	0	#fdd0a2
	wants to buy	33	0	#6baed6
Fall	could fall	45	1	#393b79
	fall	13,951	1	#637939
	fall to	245	1	#8c6d31
	fall to lowest	71	1	#7b4173
	falling	134	1	#e6550d
	falls	7,444	1	#31a354
	falls sharply	48	1	#636363
	falls to	503	1	#ff9896
	falls to lower	87	1	#1f77b4
	may fall	60	1	#aec7e8
	seen falling	73	1	#d62728
	set to fall	74	1	#e7ba52
	to fall	279	1	#fdd0a2
Plan	plan	7,870	2	#393b79
	planned	211	2	#637939
	planning	170	2	#8c6d31
	plans	6,723	2	#7b4173
	plans to acquire	154	2	#31a354
	plans to boost	77	2	#636363
	plans to buy	184	2	#ff9896
	plans to cut	101	2	#1f77b4
	plans to expand	36	2	#aec7e8
	plans to increase	57	2	#d62728
	plans to invest	128	2	#e7ba52
	plans to issue	233	2	#fdd0a2
	plans to launch	56	2	#c7e9c0
	plans to list	62	2	#e377c2
	plans to raise	316	2	#fc4e2a
	plans to sell	257	2	#8ca252
	plans to set	85	2	#ad494a